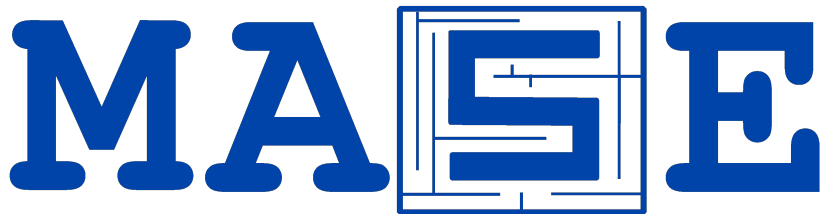


Implementation of an Eclipse-based Model Validation Plug-in for Papyrus-RT

Tuhin Kanti Das & Juergen Dingel



Modeling & Analysis in Software Engineering



Outline

- ▶ Background
- ▶ Research Stages

Analysis of UML-RT Models

- Goal: Proposing a set of design guidelines

Implementation of a Prototype Model Validator

- Goal: Providing tool support for a suitable set of the proposed

Evaluation

- Goal:
 - ☐ Providing support for the validity of the guideline catalog
 - ☐ Assessing the usefulness of the model validator

- ▶ Demo of the Prototype Plugin

Background: UML-RT

- ▶ Real time profile of UML
- ▶ Specify and implement real-time embedded software systems
- ▶ Commercial tool support: IBM Rose RT, IBM RSA RTE
- ▶ Open-source tool support: Papyrus-RT

Research Methodology

Analysis of around 100 UML-RT models

Academic Models

- Average size:
 - 5 Capsules
 - 35 States
 - 71 Transitions
 - 3 Protocols
- Maximum depth of state nesting: 4

Industrial Models

- Mostly telecommunication models
- More complicated
- Maximum depth of state nesting: 6

Discussion with UML-RT practitioners

Development guidelines for UML-RT models

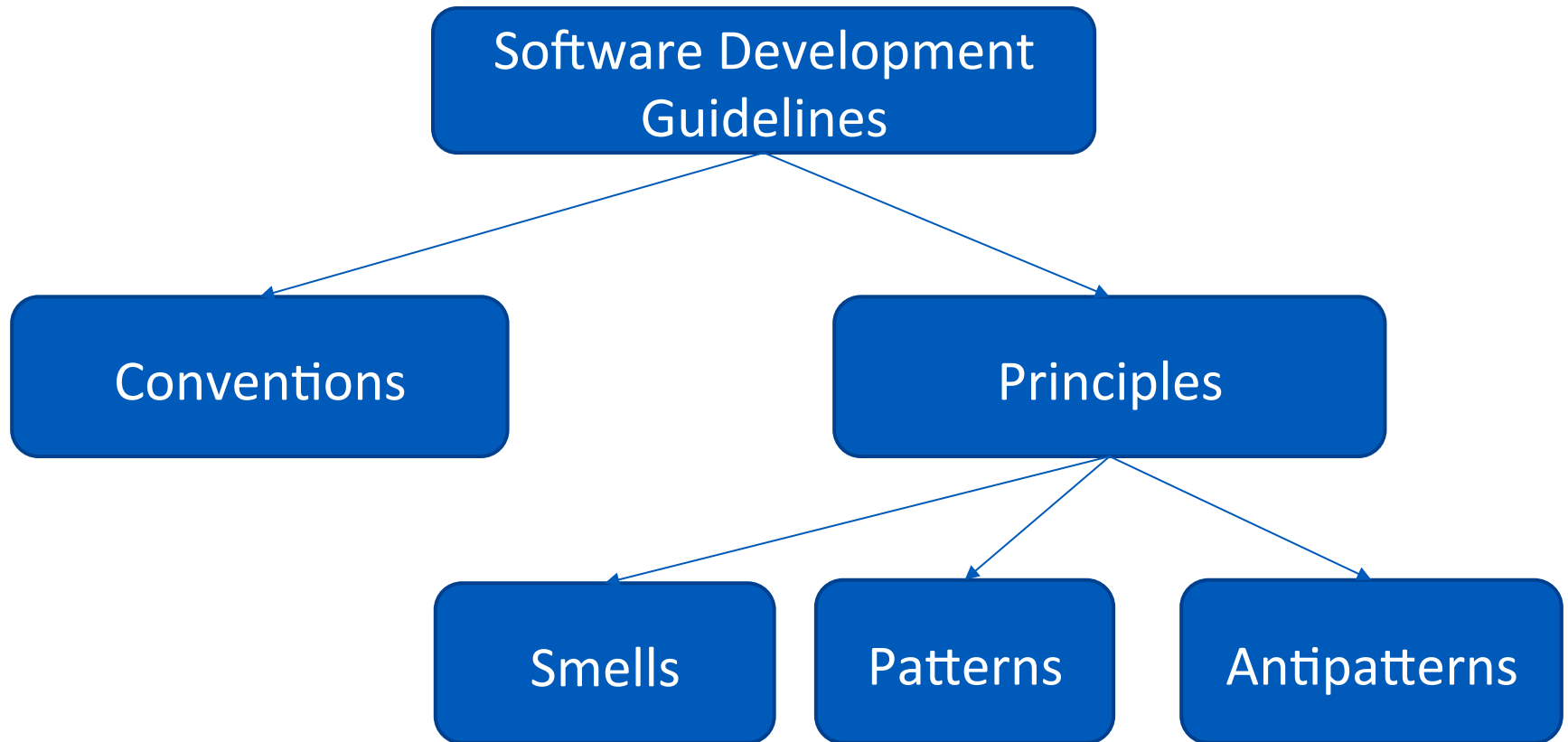


Figure: Classification of Software Development Guidelines

Proposed Design Guidelines

- ▶ Conventions: 8
- ▶ Patterns: 3
- ▶ Anti-patterns: 20

[1] T. K. Das and J. Dingel, "State machine antipatterns for UML-RT", ACM/IEEE 18th International Conference on MODELS'15 (2015).

[2] T. K. Das and J. Dingel, "Model development guidelines for UML-RT: conventions, patterns and antipatterns", Software and Systems Modeling (SoSyM) (2016).

[3] T. K. Das and J. Dingel, "Model development guidelines for UML-RT", Technical Report 2016-628, Queen's University, Canada (2016).

Proposed Design Guidelines

Conventions: 8

- ▶ Patterns: 3
- ▶ Anti-patterns: 20

Resource Utilization

- Cancellation of timers after their Use
- Termination of Created Capsules after Their Use
- Removal of Unconnected Ports
- Removal of Unreachable Artifacts

UML-RT Transitions

- Proper Use of Initial Transitions
- Use of Internal Self-Transitions instead of External

Others

- Conjugation of Server-side Ports in a Binary Protocol
- Improving the Visualization of UML-RT Diagrams

Proposed Design Guidelines

► Conventions: 8

Patterns: 3

► Anti-patterns: 20

Behavioral

- Status Monitoring Scenario
- Proper Use of System Timers

Structural

- Separation of Responsibilities

Proposed Design Guidelines

► Conventions: 8

► Patterns: 3

Anti-patterns: 20

Inspired by Fowler's Code Smells

- Refused Bequest in UML-RT
- Exclusion of Inherited Features
- Existence of Data Clumps in UML-RT
- Lack of Cohesion (Divergent Change) in UML-RT
- Feature Envy in UML-RT
- Speculative Generality in UML-RT
- Duplication in UML-RT

Choice-Point (Behavior)

- Overlapping Conditionals
- Choice-point with Non-exhaustive Guards

UML-RT Transitions (Behavior)

- Inappropriate Use of Self-Transitions
- Misuse of Local Transitions
- Incorrect Use of Group Transitions

Inheritance (Structure/Behavior)

- Absence of Inheritance
- Pitfalls of Using Promote/Demote Operations in UMLRT Inheritance

Guard Conditions (Behavior)

- Misuse of Guard Conditions
- Guards in Junction Points are affected by Action Code in Incoming Transition

Others

- Incorrect Use of Asynchronous Communication (Behavior)
- Incorrect Use of One-shot Timers (Behavior)
- Hidden States (Behavior)
- Inappropriate Modeling Scope (Structure)

Implementation: Unreachable States

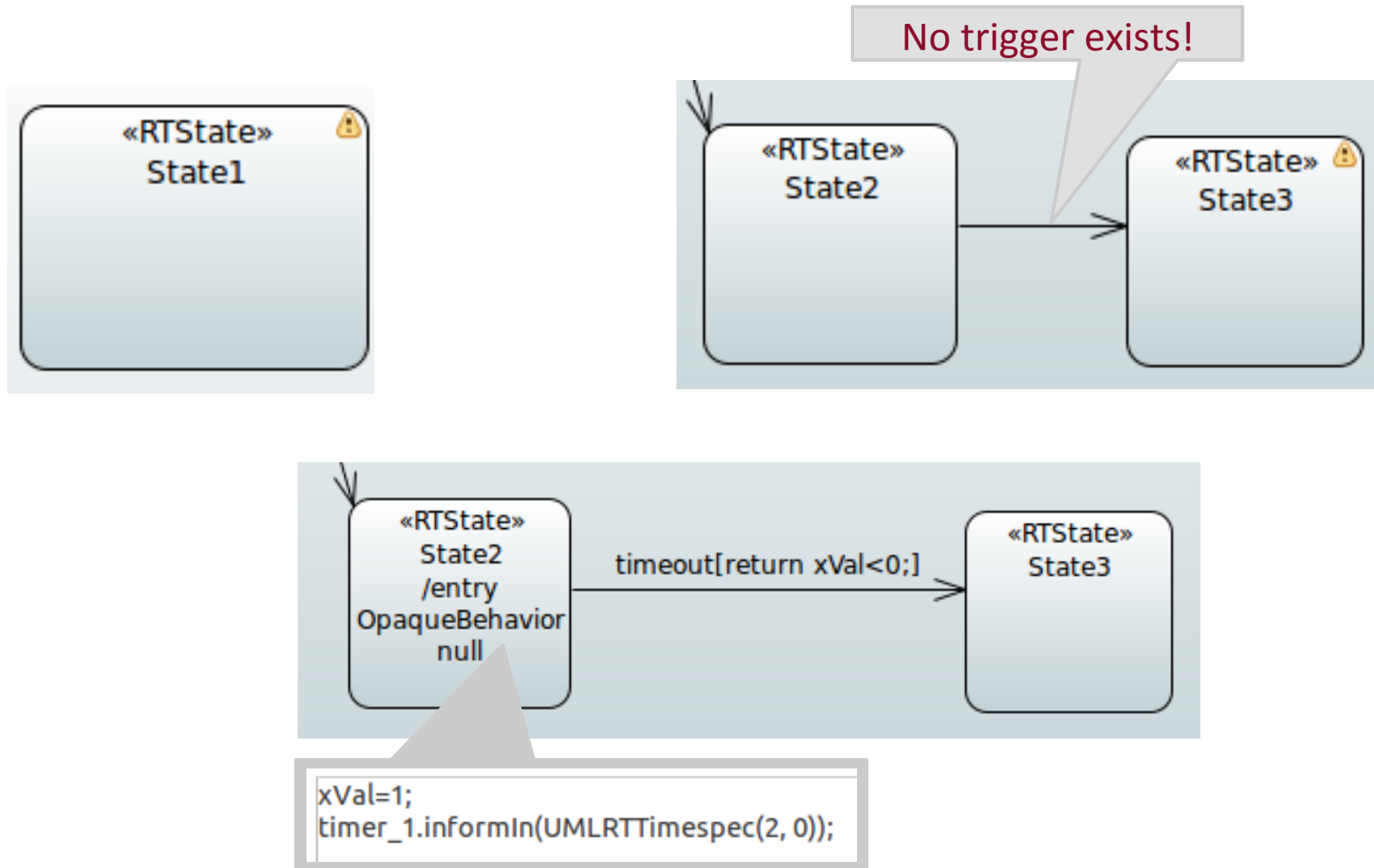


Figure: Unreachable State

Implementation: One-shot Timer as Periodic

Two Types of Timers in UML-RT

- ▶ One-shot: `timer.informIn(UMLRTTimespec(2, 0));`
- ▶ Periodic: `timer.informEvery(UMLRTTimespec(2, 0));`

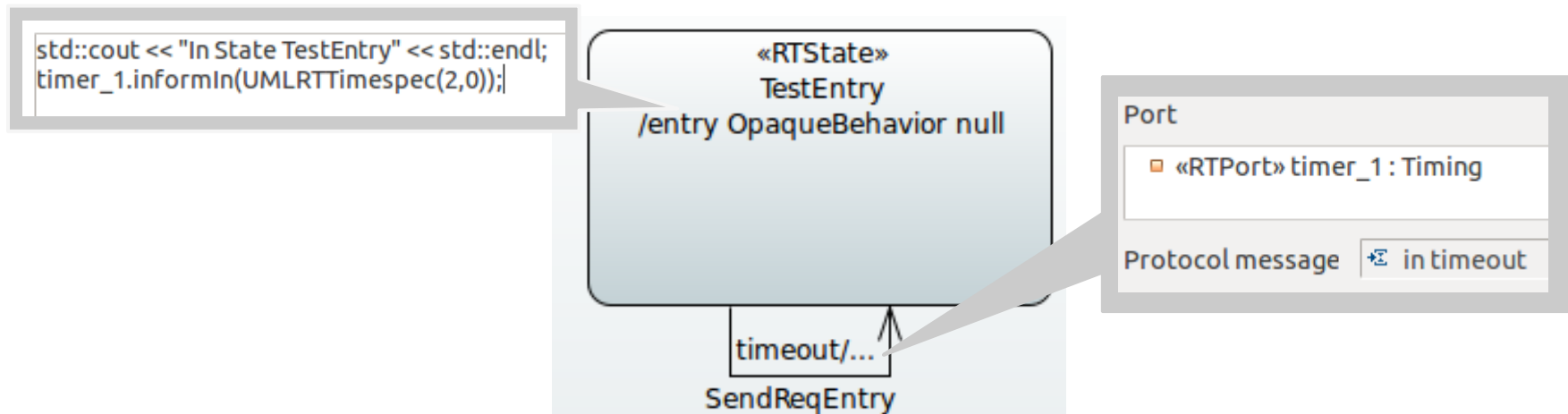


Figure: Use of One-shot Timer as Periodic

Implementation: One-shot Timer as Periodic

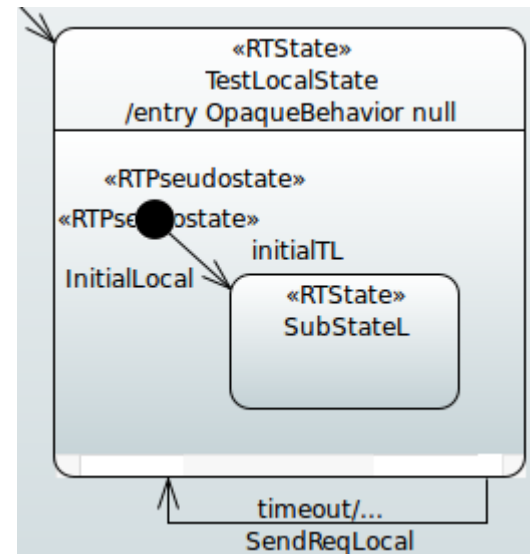
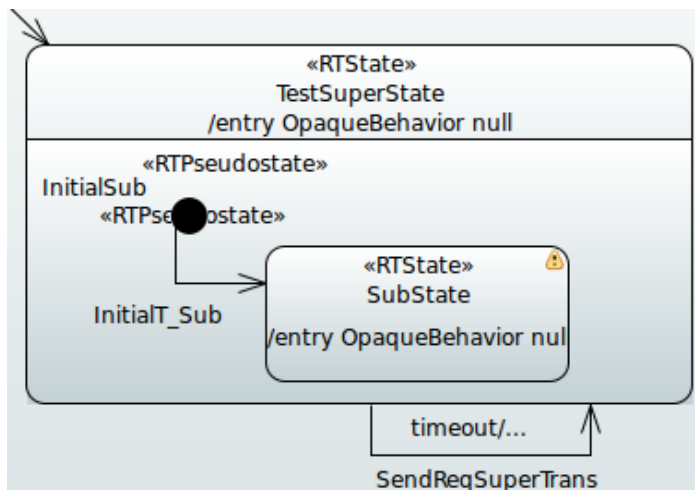
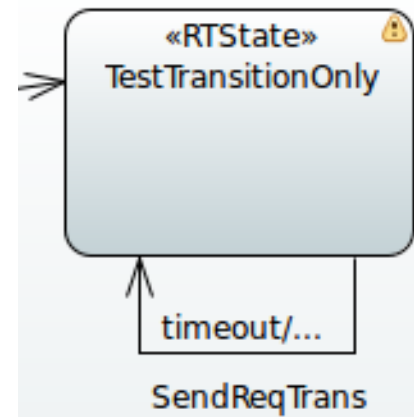
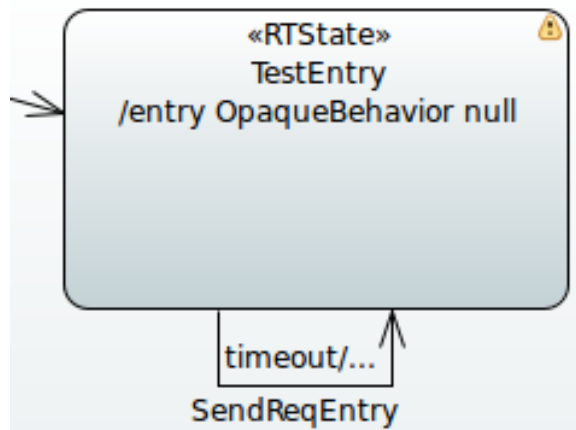


Figure: Use of One-shot Timer as Periodic

Implementation: Dependency on Requirement Specs

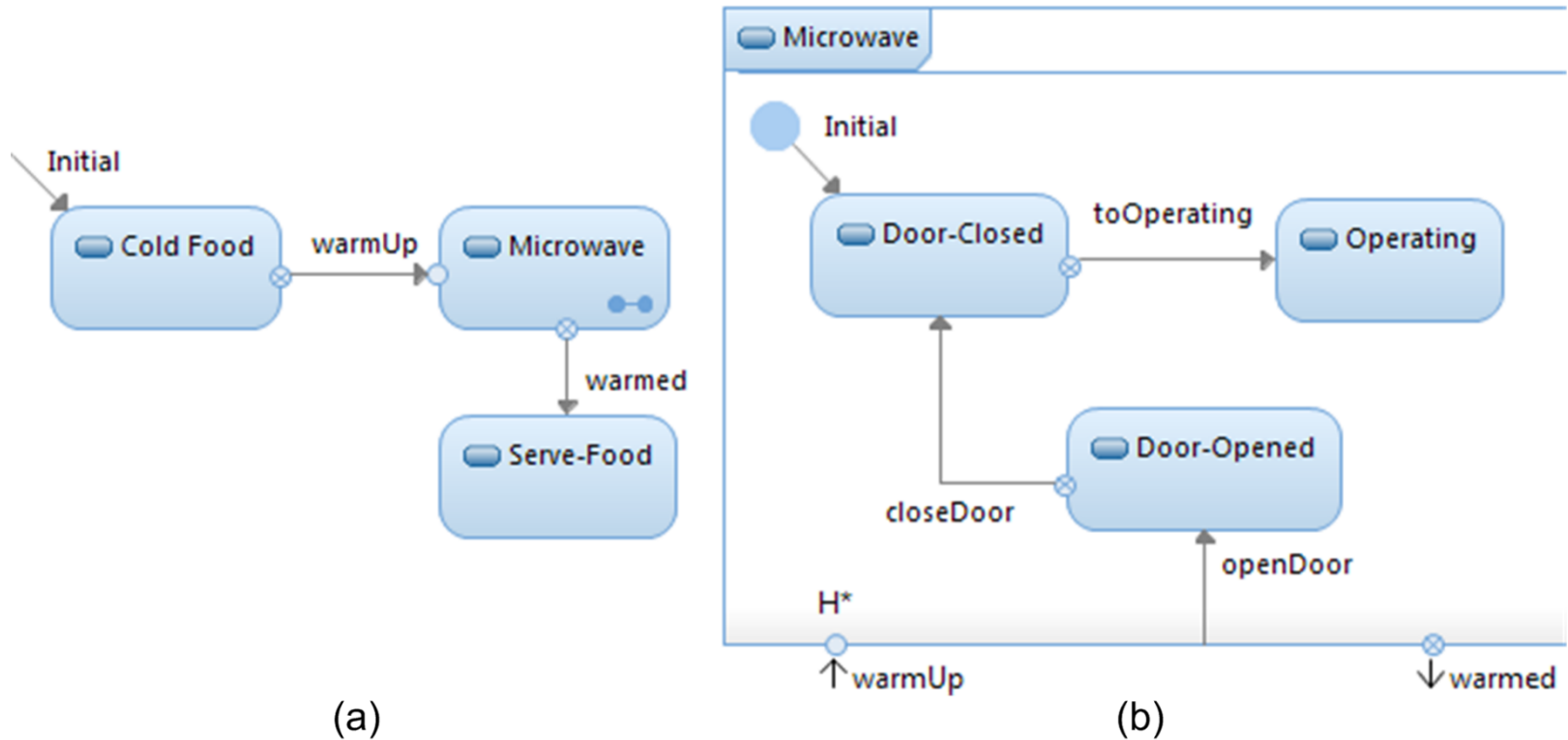


Figure: Incorrect Use of Group Transition

Implementation: Dependency on Requirement Specs

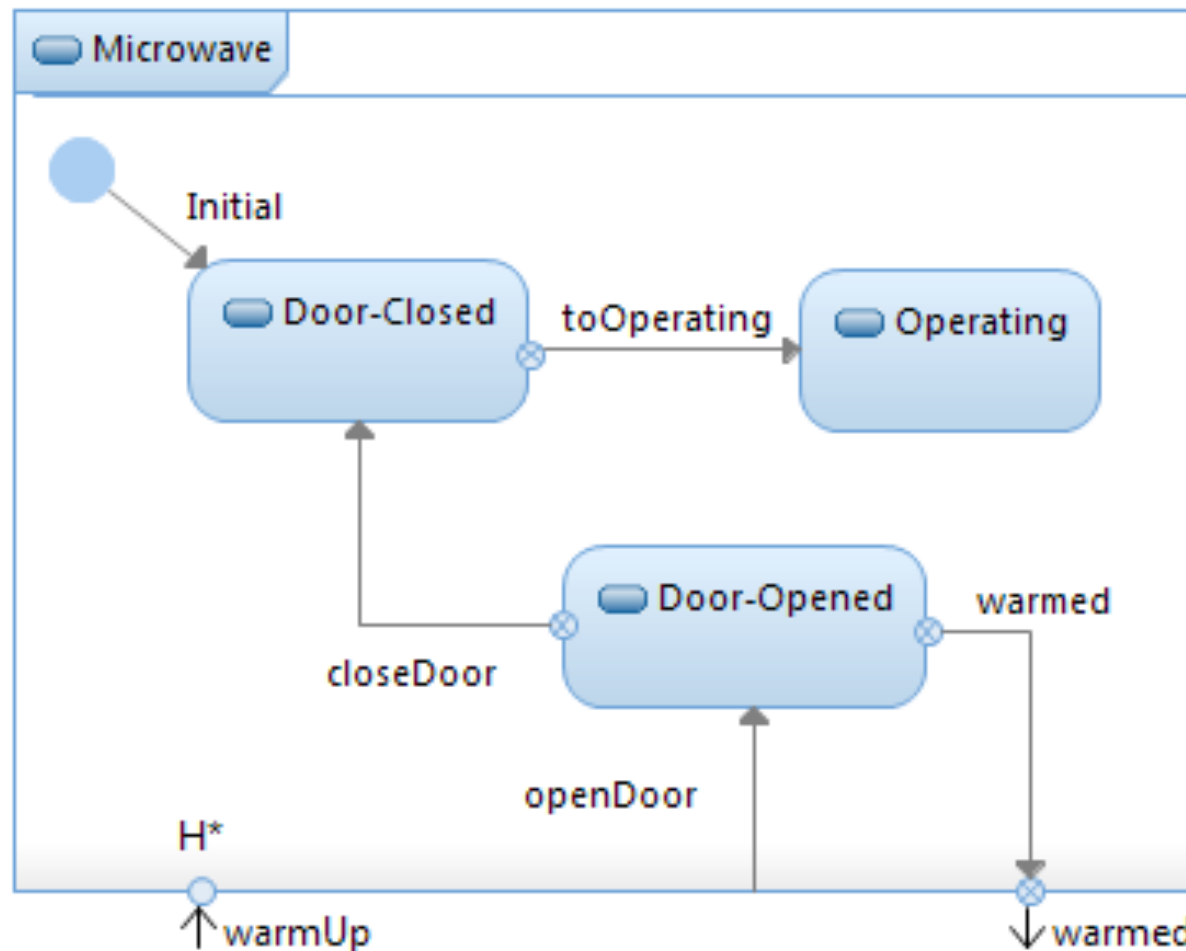
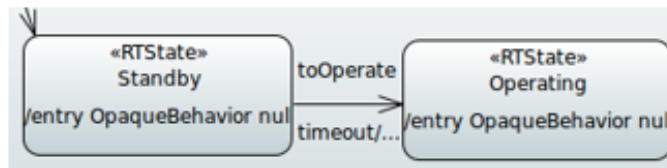


Figure: Refactored Solution for Group Transition

Evaluation Criteria

- ▶ Application of the validator in a third party model repository
- ▶ Feedback from UML-RT practitioners
- ▶ User Study: a survey
 - ▶ Direct response: Acceptance of the guidelines among the community
 - ▶ Indirect response: Problem solving queries

Q. Cancellation of timers after their use



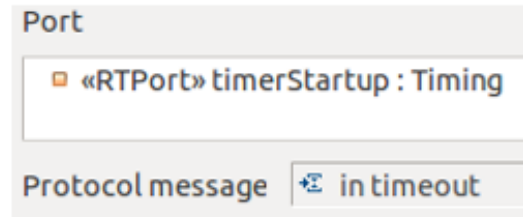
(a)

```

std::cout << "In Standby" << std::endl;
timerStartup.informIn(UMLRTTimespec(2,0));
timerStop.informIn(UMLRTTimespec(10,0));

```

(b)



(c)

Figure 1: (a) Capsule state diagram (b) Entry code of “Standby” state (c) Trigger configuration of “toOperate” transition

A timer named “timerStop” created in the action code of “toOperate” has neither been used as event trigger nor been cancelled. Consequently, the container capsule would receive a timeout event for the timer unexpectedly.

```

Controller "DefaultController" running.
In Standby
In toOperate
In Operating
Unexpected message to capsule instance Top role (no role)
on port timerStop protocol Timing signal timeout

```

Figure 2: Notification regarding unexpected timer message in command prompt

Do you think it is necessary to create a validation constraint for detecting unused timers like this?

- Ans.:**
- a. Yes, this causes waste of resources and can introduce errors if the timeout event is used undesirably
 - b. No, it is not necessary
 - c. Other... (text-box)

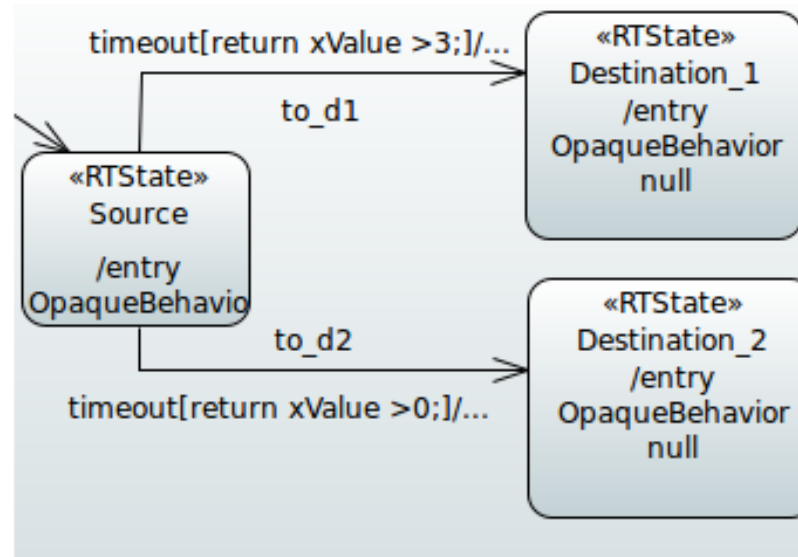
Q. Overlapping conditionals between states

Figure 1: Overlapping conditionals in two outgoing transitions of the “Source” state

In Figure 1, both of the outgoing transitions from the “Source” state are triggered by the same event, the “timeout” message of a timing port “timer_1”; however, they have different guards associated with them which decides the execution behavior. Assuming “Source” as the currently active state and “xValue=5” in the “Source” state, which state will get executed if the capsule receives a timeout message from “timer_1”?

- Ans.:**
- a. “Source” state
 - b. “Destination_1” state
 - c. “Destination_2” state
 - d. Other.. (text-box)

Please Help Evaluate the Research!

- ▶ You will receive an email once the survey is online
- ▶ Approximate duration: 15-20 minutes

Evaluation of Model Validation Plug-in for Papyrus-RT Subscription form to participate in the survey

Please provide your email contact below to subscribe in the survey. The survey will be used for evaluating the model validation plug-in for Papyrus-RT. Once the survey is online, we will send you a webpage link for participating in the survey. The approximate time for completing the survey would be 15-20 minutes. Your responses will be collected anonymously. Thank you very much for your great support!

ID	Email address
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	

Current State of the Prototype Validator

- ▶ DEMO

Thank you!